LDJ

CLMPTO

02/07/03

1. (Twice Amended) A processor comprising:

a register file; and

a functional unit, coupled to the register file, that executes an instruction that operates upon plural registers of said register file, including at least one register explicitly identified by an explicitly defined register specifier and at least one other register implicitly identified by the explicitly-defined register specifier.

Claim 2 is cancel.

3. (Amended) A processor according to Claim 1 wherein:

a register specifier for the other register is implicitly derived by adding one to the explicitly-defined register specifier.

4. A processor according to Claim 1 wherein the processor is a Very Long Instruction Word (VLIW) processor and wherein

said register file further including a plurality of register file segments wherein the plurality of registers are divided among said plurality of register file segments; and

wherein said VLIW processor further comprises a plurality of functional units, ones of the plurality of functional units being coupled to and associated with respective ones of the register file segments.

5. A processor according to Claim 1 wherein:

the instruction is a multiply-add instruction that uses an implicitly-derived register specifier and has a form of:

muladd rsl, rs2, rd,

and performs an operation specified by the equation:

rd = (rsl * [rsl+l]) + rs2,

where the term [rsl+1] designates data contained within the register following the explicitly-defined register rsl.

6. (Amended) A processor according to Claim 1 wherein:
the instruction is a bit extract instruction that uses an implicitly-derived register specifiers and has a form of:

bitext rsl, rs2, rd,

and per.forms an operation of extracting bits from even-aligned pairs of registers r[rsl] and [rsl+ 1] where the term [rsl+l] designates data contained within the other register following the explicitly-defined register rsl, wherein data in register r[rs2] describes the extracted field of registers r[rsl] and r[rsl + 1] and register r[rd] is a destination register.

7. (Amended) A processor according to Claim 1 wherein:

the instruction is a call instruction that uses an implicitly-derived register specifiers and has a form of:

call label,

causing a control transfer to an address specified by a label operand, the address of the instruction word following the instruction word begun with the call instruction is held in an alias register, an implicit operand of the call instruction, with an assembler using an alias link pointer lp for pointing to the alias register.

8. A processor according to Claim 1 wherein:

the instruction is a double-precision floating point add instruction that uses an implicitly-derived register specifiers and has a form of:

dadd rsl, rs2, rd,

and performs an operation specified by the equation:

(rd, [rd+1] =(rsl, [rsl+l])+(rs2, [rs2+1]),

where the terms (rsl, [rsl+1]), (rs2, [rs2+1]), and (rd, [rd+l]) designate double-precision words.

9. (Amended) A processor according to Claim 1 wherein:

the instruction is a double-precision floating point compare instruction that uses an implicitly-derived register specifiers and has a form of:

dcmpcc rsl, rs2, rd,

and performs an operation of comparing data in registers (rsl, [rsl+1]) with data in registers (rs2, [rs2+1]) and storing a result in registers (rd,[rd+l]) where the terms (rsl, [rsl+l]), (rs2, [rs2+1]), and (rd, [rd+1]) designate double-precision words, and cc designates a condition code including equal, less than, and less than or equal to conditions.

10. A processor according to Claim 1 wherein:

the instruction is a double-precision floating point multiply instruction that uses an implicitly-derived register specifiers and has a form of:

dmul rs1, rs2, rd,

and performs an operation specified by the equation:

(rd, [rd+1] =(rsl, [rsl+1]) * (rs2, [rs2+1]),

where the terms (rsl, [rsl+1]), (rs2, [rs2+l]), and (rd, [rd+1]) designate double-precision words.

11. (Amended) A processor according to Claim 1 wherein:

the instruction is a double-precision floating point subtraction instruction that uses an implicitly-derived register specifiers and has a form of:

dsub rsl, rs2, rd,

and performs an operation specified by the equation:

(rd, [rd+l]) = (rsl, [rsl+l]) - (rs2, [rs2+1]),

where the terms (rsl, [rsl+l]), (rs2, [rs2+1]), and (rd, [rd+1]) designate double-precision words.

12. A processor according to Claim 1 wherein:

the instruction is a pack instruction that uses an implicitly-derived register specifiers and has a form of:

pack rsl, rs2, rd,

and operates upon a register pair (rsl, [rsl+l]) as four signed 16-bit operands, and shifts the four operands right by a value designated by the register specified by rs2, clips the shifted 16-bit operands within defined limits and stores the clipped 16-bit operands in a register pair (rd, [rd+1]).

13. A processor according to Claim 1 wherein:

the instruction is a double-precision floating point conversion instruction that uses an implicitly-derived register specifiers and has a form of:

dtox rsl, rd,

and performs an operation of converting a double-precision floating point value to a specified format x, the format x including a single-precision floating point format (dtof), an integer format (dtoi), and a long integer format (dtol).

14. A processor according to Claim 1 wherein:

the instruction is a double-precision floating point absolute value instruction that uses an implicitly-derived register specifiers and has a form of:

dabs rsl, rd,

and performs an operation of converting a double-precision floating point value in a register pair (rsl, [rsl+1]) to an absolute magnitude in a register pair (rd, [rd+1]).

15. A processor according to Claim 1 wherein:

the instruction is a double-precision floating point negative value instruction that uses an implicitly-derived register specifiers and has a form of:

dneg rs1, rd,

and performs an operation of converting a double-precision floating point value in a register pair (rs1, [rs1 +1 ]) to a negative magnitude in a register pair (rd, [rd+1]).

16. A processor according to Claim 1 wherein:

the instruction is a double-precision floating point set limit instruction that uses an implicitly-derived register specifiers and has a form of:

dlim rsl, rs2, rd,

and performs an operation of setting a double-precision destination register (rd, [rd+1]) to the maximum of a double-precision first source register (rsl, [rsl+l]) and a double-precision second source register (rs2, [rs2+1]), or setting the double-precision destination register (rd, [rd+l]) to the minimum of a double precision first source register (rsl, [rsl+l]) and a double precision second source register (rs2, [rs2+1]),

where the terms (rsl, [rsl+l]), (rs2, [rs2+1]), and (rd, [rd+1]) designate double-precision words.

17.(Amended) A processor according to Claim 1 further comprising:
a decoder coupled to the functional unit and configured to generate a first pointer pointing to the explicitly-specified register and a second pointer pointing to the other register.

Claim 18 is cancel.

19.(Amended) A processor according to Claim 1 further comprising:
a pointer coupled to the register file and designating a register in the register file, the pointer including a signal indicative of selection of a the other register, wherein a register read of the explicitly-specified register is accompanied by a register read of the other register when implicit derivation of the explicitly-defined register specifier is selected.

20. (Twice Amended) A method of operating a processor comprising:
storing information in a register file including a plurality of registers;
executing instructions in a functional unit coupled to the register file and operating upon a plurality of registers in the register file;
explicitly defining a register specifier of a register operated upon during executing of the instruction; and
implicitly deriving a register specifier of at least one other register operated upon during executing of the instruction based on the explicitly defined register specifier.

21. (Amended) A method according to Claim 20 further comprising:
decoding an instruction; and
deriving, during decoding of the instruction, a register specifier based on the explicitly-specified register specifier of the instruction.

Claim 22 is cancel.

23. (New) The processor of claim 1, wherein the register specifier is encoded as an indirect specifier.

24. (New) A method according to Claim 20 further comprising:
implicitly deriving the register specifier for the other register by adding one to the explicitly-defined register specifier.